Tracked Changes: Navigating the Document-Format Anti-Pattern

Dennis E. Hamilton
Interoperability Architect
4401 44th Ave SW
Seattle, WA 98116, USA
+1-206-779-9430
dennis.hamilton@acm.org

ABSTRACT
Editing of word-processing documents at the presentation level, with visible tracking of changes, operates at a different level of abstraction and granularity than representation of the document in common document-file formats. The consequent mismatches are demonstrated using OpenDocument format provisions for tracked changes. A selection-copy analogy is introduced for bridging the abstraction levels while adhering to file-format provisions. The enhancements improve reliability and interoperability and are implementable incrementally without obsoleting current software and documents.

Categories and Subject Descriptors
H.4.1 [**Information Systems Applications**]: Office Automation – *word processing;* D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement – *portability; restructuring, reverse engineering, re-engineering;* D.2.9 [**Software Engineering**]: Management – *life cycle;* H.1.2 [**Models and Principles**] User/Machine Systems – *human factors,* I.7.1 [**Document and Text Processing**]: Document and Text Editing – *version control, document management*

General Terms
Algorithms, Design, Management, Standardization, Verification

Keywords
OpenDocument, XML, change tracking, document formats, overlapping markup, user conceptualization, WYSIWYG.

1. INTRODUCTION
Repaired Change Tracking (RCT) is an extension profile for change-tracking provisions in OpenDocument format (ODF)[8].

RCT accounts for the level of abstraction that users perceive when operating with graphical word-processing software. The clash and mismatch with representation in standard file formats is part of the situation to be comprehended (section 2). The disparity is evident in the format-level representation of tracked changes (section 3). Reconciliation of the disparity sufficient for improved interchange follows (section 4).

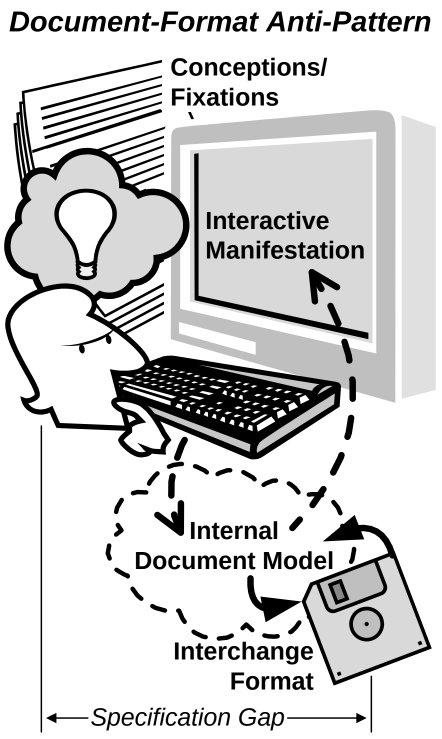


Figure 1. WYSIWYG Document Processing Interactions (typical). Software implements an internal document model in order to manifest views manipulated at the graphical user interface, shielding users from the intricacies of interchange formats. Microsoft Word, OpenOffice.org Writer, and other word-processing programs exhibit this pattern.

2. THE SITUATION
In today's office-productivity software suites, document production is accomplished by manipulations of a graphical-interface presentation, a *manifestation*, of the document under development or review. The operator initiates, manipulates and reviews the manifestation as the document develops. Success is achieving a manifestation having acceptable appearance of the content, reinforced by obtaining a faithful printed edition or some final electronic form (Figure 1). Through these interactions, users are trained to their tools through trial and error.

Software that operates the manifestation and interaction process maintains, in some manner, an internal document model that captures what there is of the document at any point in time.

The software also saves work in one or more file formats for digital interchange of editable documents. Commonly-employed formats include ODF[7], Office Open XML[3], and older formats such as those of Microsoft Office[5].

It is important to appreciate that the emitted file formats are not directly used for authoring, editing, or any purpose but re-introduction into computer programs that manifest the represent-ed document. It is near-inconceivable that users of the software have either interest or means to interpret complex document files and discern their relationship to the recognizable documents that are represented.

The disparate levels at which documents are manifest to users and represented in interchange formats is a consequence of the *document-format anti-pattern*[1]: Specifications for these docu-ment formats do not address manifestation; the formal require-ments and definitions are not related to any internal model. The anti-pattern's absence of specified behavior is intentional (Figure 1).

Consequently, achievement of interchange fidelity and interoper-ability is driven by forces and agreements between parties that are beyond the scope of the file-format specifications.

That is the setting in which interoperable change-tracking is expected.

3. CHANGE-TRACKING COMPLEXITIES

ODF employs XML documents as carriers of the structure and content of the represented OpenDocument document. Represen-tation of ODF features and structure introduces off-hierarchy interdependencies in the XML[1]. Structural-consistency and referential-integrity constraints for the OpenDocument document representation are not conveyed in the ODF XML schema ([8] Appendix A).

The anti-pattern and complexities of XML representation are magnified in the case of change-tracking provisions ([8] section 5.5).

3.1 Tracked-Deletion Representation

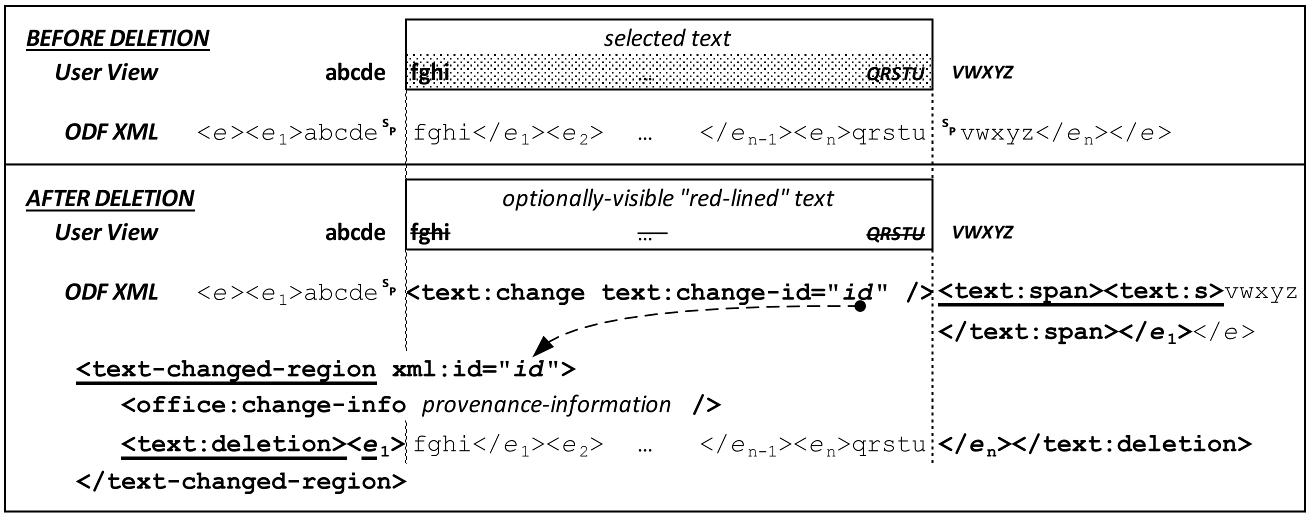
For deletions, typical manifestations present a progression of formatted text in some progressive layout. Users select spans of individual visual elements by some means. The deletions are achieved by single actions (Figure 2).

After a deletion the document appears to have a new continuous progression with the selected material absent. The textual material on each side of the deletion usually appears unchanged. There may be layout effects, including material beyond the deletion drawn into the layout of material preceding the deletion.

When changes are tracked and presented for review, deleted material will be presented as it was, along with some visible indication of deletion, such as colorization and strike-through.

In the XML representation, deletions may be far more conse-quential than what is manifest. Multiple XML elements of the representation can be excised. Other elements may have been severed at the edges of the deletion, with start tags losing their end tags and *vice versa*. Although the deletion can be represented by a single deletion point, an *invisible seam*, cosmetic *curing*

OpenDocument Tracked-Deletion Pattern (simplified)



```
BEFORE:  <e><e1>abcde sp fghi</e1><e2>  …  </en-1><en>qrstu sp vwxyz</en></e>

AFTER:   <e><e1>abcde sp <text:change text:change-id="id" /><text:span><text:s>vwxyz
         </text:span></e1></e>

  <text-changed-region xml:id="id">
     <office:change-info provenance-information />
     <text:deletion><e1>fghi</e1><e2>  …  </en-1><en>qrstu</en></text:deletion>
  </text-changed-region>
```

Figure 2. Cross-cutting text deletions replace excised text and markup components with user-invisible seams. The seam element replaces extracted material set-aside elsewhere in the XML representation. Curing occurs beyond the seam in order to have a properly rendered result, as in bold-italic small caps rendering of $^{s}{}_{p}$vwxyz. Orphaned end tags for elements whose start tags are consumed in the deletion (</e$_n$> in the figure) are cured by tying to widowed (or new) start tags as appropriate. Adjoined white space ($^{s}{}_{p}$ in the XML) is preserved as seen before deletion. Set-aside material is "wrapped" with XML tags to be well-formed in place, with introduction of namespace bindings as needed in the scope of the set-aside location[2]. RCT introduces extended attributes in start tags (underlined in the figure).

[1]http://en.wikipedia.org/w/index.php?title=Anti-pattern&oldid=612745076

may be necessary adjacent to the seam. Curing adjusts surrounding XML tags and introduces additional elements so that the document is manifest as if there is no disturbance to text content adjacent to the deletion[2]. Deletion is not a context-free activity at the XML representation level.

Tracking and reversion of deletions require retention of the extracted material. When XML elements are severed, XML start tags and end tags are adjoined to the extraction, achieving a valid XML `<text:deletion>` element (Figure 2).

3.2 Cross-Cutting Markup Effects

Three XML elements are seams in the ODF representation of tracked changes. Along with `<text:change />` for deletions, `<text:change-start />` and `<text:change-end />` bracket insertions. Traversal of XML from a `-start` seam to its corresponding `-end` seam can be off-hierarchy, cross-cutting the XML akin to the deletion case. All seam elements identify an out-of-line `<text:changed-region>` element providing particulars about the kind of change, as in (Figure 2).

There are also off-hierarchy bracketing's for non-change-tracking purposes (e.g., annotation and phrase indexing) in ODF's XML representation. Any bracket pairing, including around an insertion, can be severed by a deletion. Insertions can spread bracket pairs farther apart. Deletions can also capture markers that serve as targets, as sources for cross-references, and connections that bind structural features of the ODF document, breaking the associations.

3.3 Atomicity Conflicts and Failures

In ODF 1.2, all changes are represented as unconnected insertions and deletions. Substitution by insertion into a selection becomes a deletion following by an insertion. If a selection is moved from another part of the document, there is no association of the insertion with deletion at the original location. Failure to treat these actions as joined and atomic leads to conflicts when a deletion is reverted and an associated insertion is accepted. This is exacerbated when implementations divide single-selection deletions and insertions into multiple, smaller change-tracked operations, presenting users with inscrutable side-effects of their actions.

Copying of certain selections, if allowed, introduces conflicts in the document structure, including violation of internal identifier uniqueness in the XML[4]. The outcome can be irreparable loss of document structural connections, including inbound identifiers, in the XML representation of the changes in the file.

4. RCT APPROACH

The fundamental extension to accomplish RCT consists of additional attributes in the out-of-line `<text:changed-region>` elements and their change-specific sub-elements (as in Figure 2). New attributes provide details for correctly-reversing cures introduced at seams. Other attributes chain `<text:changed-region>` elements together when they are intended to be taken as part of a single atomic change action, with acceptance or reversal as a whole. The links can also be used to assert partial ordering among overlapping and colliding changes so that incorrect order of acceptance and reversal is inhibited.

The selection-copy analogy relates occurrences of tracked-change seams and associated `<text:changed-region>` elements to hypothetical selection and copy operations at the manifestation level. A changed region is expected to be tied to a selection in the manifestation, there being limited places to point and select in a practical manifestation. Implementations need only manifest those changed regions of an input OpenDocument document that the implementation could have produced. Implementations may ignore RCT features by design and when they are not representations an implementation is designed to support. The default is to treat the tracked changes as accepted.

Selections of sources for copies and/or a move can be further characterized as if the selections are extracted into XML markup akin to set-aside deletion markup, even when no deletion has occurred. Insertion of such a copy is characterized as the reversion of a deletion that was already at the place of insertion, although different curing may be involved in introducing the insertion. Explicit representation of a selection as an XML element in this manner is also practical for delivery by clip-board negotiation into a different document or embedding via OLE[6].

5. OUTLOOK

Development of RCT starts with comprehensive analysis of the ODF schema and specification. All interdependencies and interactions between tracked-change markup and other markup of ODF documents are identified. A stable set of generic cases is chosen by which combinatorial explosion is avoided and tracked-change overlaps and collisions are addressed.

Generic cases are expressed in terms of selection, deletion, insertion, copying, moving, and curing at change boundaries.

Description is at the level of XML representation of the changed document and of clip-board equivalents. There is no appeal to internal document models and interactive manifestations beyond the selection-copy analogy.

Analysis, collection of details, derivation of principles and guidance are maintained on a public web site.[3]

6. CONCLUSIONS

Any approach to recorded and presented change-tracking of ODF-based documents must reconcile the same cases identified for RCT. The RCT analysis is valuable support to any efforts of greater ambition.

Three features of OpenDocument representation secure the opportunity for repair and improvement of textual change-tracking via RCT: (1) empty-element XML markers where changes have been made, (2) correct final form when markers are ignored, and (3) extension attributes ignorable by default.

[2] Since the seam is invisible in the rendering of text contiguous with it, there are exotic cases where appearance does change depending on altered character positions in words and applicable text-joining rules[9]. Visualizations of tracked changes might present these cases as substitutions so the situation is made explicit.

[3] http://nfoworks.org/notes/2014/05/n140501.htm

By selection-copy analogy, RCT avoids assumption of particular manifestations of tracked changes, of internal document models, and of implementation details.

Ultimately, deployment depends on the difficulty of adjusting existing implementations to introduce and refine the change-tracking that is produced, recognized, and, where not supported, appropriately ignored.

The devil is in the details.

7. ACKNOWLEDGMENT

Thanks to colleague William L. Anderson for his enthusiastic attention and careful eye.

8. REFERENCES

[1] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau (editors). .*Extensible Markup Language (XML) 1.0 (Fourth Edition)*. W3C Recommendation 16 August 2006, edited in place 29 September 2006. http://www.w3.org/TR/2006/REC-xml-20060816.

[2] Tim Bray, Dave Hollander, Andrew Layman, and Richard Tobin (editors). *Namespaces in XML 1.0 (Second Edition)*. W3C Recommendation 16 August 2006. http://www.w3c.org/TR/2006/REC-xml-names-20060816.

[3] Ecma International. *Office Open XML File Formats*, 4th edition. Standard ECMA-376, December 2012. http://www.ecma-international.org/publications/standards/Ecma-376.htm

[4] Jonathan Marsh, Daniel Veillard, and Norman Walsh (editors). *xml:id Version 1.0*. W3C Recommendation 9 September 2005. http://www.w3c.org/TR/2005/REC-xml-id-20050909.

[5] Microsoft Corporation. Office File Formats. Microsoft Developer Network MSDN Library. Accessed 2014-06-19 at http://msdn.microsoft.com/en-us/library/cc313118(v=office.12).aspx

[6] Microsoft Corporation. OLE Background. Microsoft Developer Network MSDN Library. Accessed 2014-06-18 at http://msdn.microsoft.com/en-us/library/19z074ky.aspx

[7] OASIS. *Open Document Format for Office Applications (OpenDocument) Version 1.2*. 29 September 2011 OASIS Standard. master document introducing further parts by reference, http://docs.oasisopen.org/office/v1.2/os/OpenDocument-v1.2-os.html

[8] OASIS. *Open Document Format for Office Applications (OpenDocument) Version 1.2 Part 1: OpenDocument Schema*. 29 September 2011 OASIS Standard. Available at http://docs.oasis-open.org/office/v1.2/os/

[9] The Unicode Consortium. The Unicode Standard Version 5.2.0 defined by: *The Unicode Standard, Version 5.2* (Mountain View, CA: The Unicode Consortium, 2009. ISBN 978-1-936213-00-9). http://www.unicode.org/versions/Unicode5.2.0/